

Intel® Advisor

Vectorization Optimization and Thread Prototyping

Munara Tolubaeva
Software Technical Consulting Engineer



Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2016, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Data-Driven Threading Design

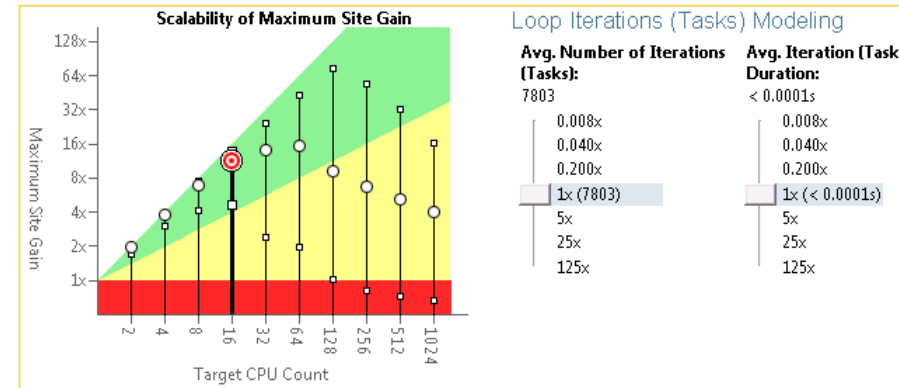
Intel® Advisor – Thread Prototyping

Have you:

- Tried threading an app, but seen little performance benefit?
- Hit a “scalability barrier”? Performance gains level off as you add cores?
- Delayed a release that adds threading because of synchronization errors?

Breakthrough for threading design:

- Quickly prototype multiple options
- Project scaling on larger systems
- Find synchronization errors before implementing threading
- Separate design and implementation - Design without disrupting development



**Add Parallelism with Less Effort,
Less Risk and More Impact**

<http://intel.ly/advisor-xe>

Data Driven Vectorization Design

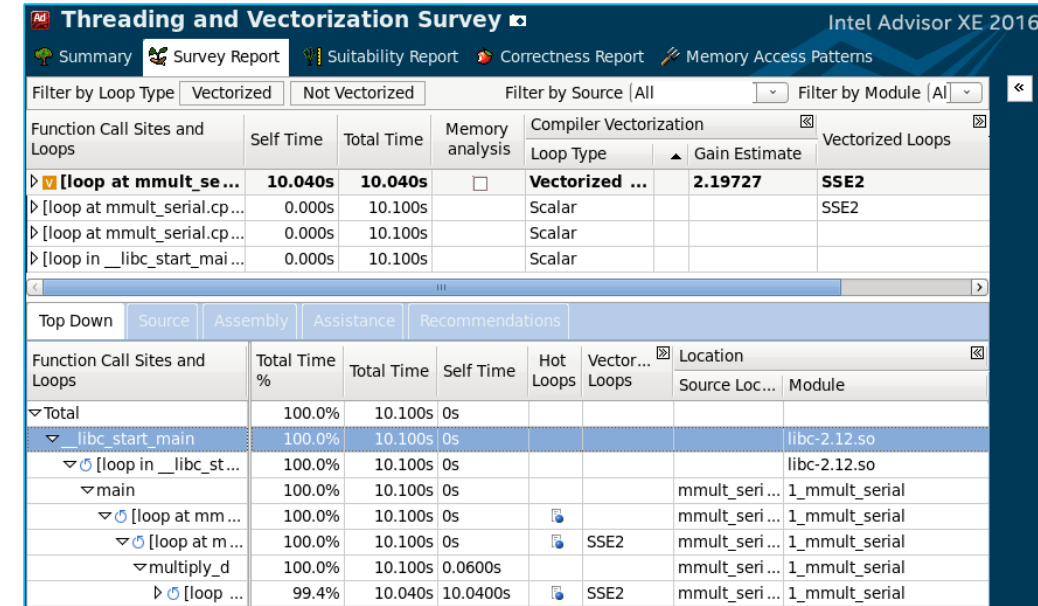
Intel® Advisor – Vectorization Advisor

Have you:

- Recompiled with AVX2, but seen little benefit?
- Wondered where to start adding vectorization?
- Recoded intrinsics for each new architecture?
- Struggled with cryptic compiler vectorization messages?

Breakthrough for vectorization design

- What vectorization will pay off the most?
- What is blocking vectorization and why?
- Are my loops vector friendly?
- Will reorganizing data increase performance?
- Is it safe to just use pragma simd?



Threading and Vectorization Survey Intel Advisor XE 2016

Summary Survey Report Suitability Report Correctness Report Memory Access Patterns

Filter by Loop Type: Vectorized Not Vectorized Filter by Source: All Filter by Module: All

Function Call Sites and Loops	Self Time	Total Time	Memory analysis	Compiler Vectorization	Gain Estimate	Vectorized Loops
▶ [loop at mmult_se ...]	10.040s	10.040s	<input type="checkbox"/>	Vectorized ...	2.19727	SSE2
▶ [loop at mmult_serial.cp ...]	0.000s	10.100s		Scalar		SSE2
▶ [loop at mmult_serial.cp ...]	0.000s	10.100s		Scalar		
▶ [loop in __libc_start_mai ...]	0.000s	10.100s		Scalar		

Top Down Source Assembly Assistance Recommendations

Function Call Sites and Loops	Total Time %	Total Time	Self Time	Hot Loops	Vector...	Location
						Source Loc... Module
▼ Total	100.0%	10.100s	0s			
▼ __libc_start_main	100.0%	10.100s	0s			libc-2.12.so
▼ [loop in __libc_st ...]	100.0%	10.100s	0s			1_mmult_serial
▼ main	100.0%	10.100s	0s			mmult_se ... 1_mmult_serial
▼ [loop at mm ...]	100.0%	10.100s	0s	<input checked="" type="checkbox"/>		mmult_se ... 1_mmult_serial
▼ [loop at m ...]	100.0%	10.100s	0s	<input checked="" type="checkbox"/>	SSE2	mmult_se ... 1_mmult_serial
▼ multiply_d	100.0%	10.100s	0.0600s	<input checked="" type="checkbox"/>	SSE2	mmult_se ... 1_mmult_serial
▶ [loop ...]	99.4%	10.040s	10.0400s	<input checked="" type="checkbox"/>	SSE2	mmult_se ... 1_mmult_serial

More Performance
Fewer Machine Dependencies

Factors that prevent Vectorizing your code

1. Loop-carried dependencies

```
DO I = 1, N
  A(I + M) = A(I) + B(I)
ENDDO
```

3. Loop structure, boundary condition

```
struct _x { int d; int bound; };

void doit(int *a, struct _x *x)
{
  for(int i = 0; i < x->bound; i++)
    a[i] = 0;
}
```

1.A Pointer aliasing (compiler-specific)

```
void scale(int *a, int *b)
{
  for (int i = 0; i < 1000; i++)
    b[i] = z * a[i];
}
```

4 Outer vs. inner loops

```
for(i = 0; i <= MAX; i++) {
  for(j = 0; j <= MAX; j++) {
    D[j][i] += 1;
  }
}
```

2. Function calls (incl. indirect)

```
for (i = 1; i < nx; i++) {
  x = x0 + i * h;
  sumx = sumx + func(x, y, xp);
}
```

5. Cost-benefit (compiler specific..)

And others.....

Factors that **slow-down** your **Vectorized** code

1.A. Indirect memory access

```
for (i=0; i<N; i++)  
    A[B[i]] = C[i]*D[i]
```

1.B Memory sub-system Latency / Throughput

```
void scale(int *a, int *b)  
{  
    for (int i = 0; i < VERY_BIG; i++)  
        c[i] = z * a[i][j];  
        b[i] = z * a[i];  
}
```

2. Small trip counts not multiple of VL

```
void doit(int *a, int *b, int  
unknown_small_value)  
{  
    for(int i = 0; i <  
unknown_small_value; i++)  
        a[i] = z*b[i];  
}
```

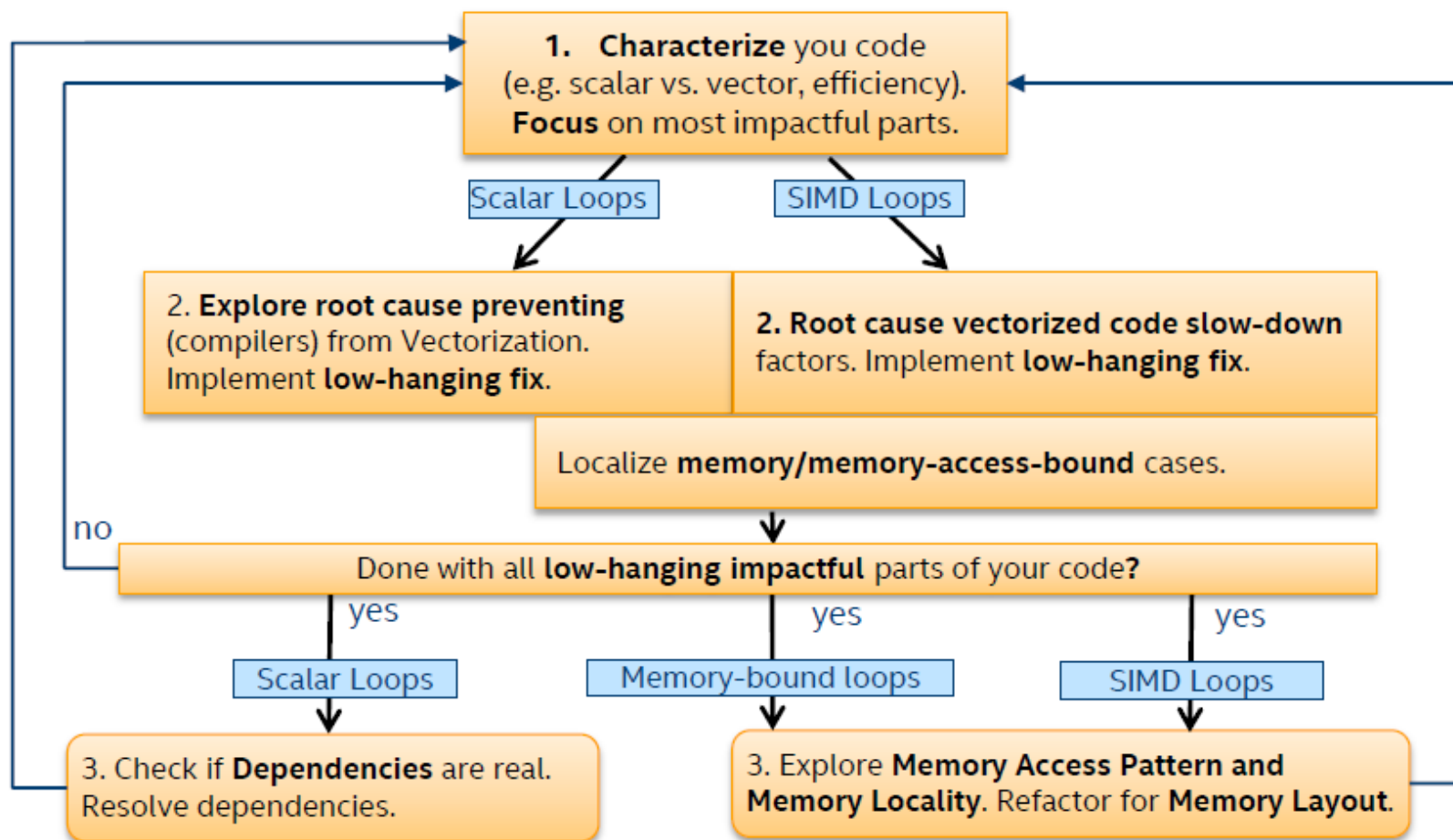
3. Branchy codes, *outer vs. inner loops*

```
for(i = 0; i <= MAX; i++) {  
    if ( D[i] < N)  
        do_this(D);  
    else if (D[i] > M)  
        do_that();  
    //...  
}
```

5. **MANY** others: spill/fill, fp accuracy trade-offs, FMA, DIV/SQRT, Unrolling, even AVX throttling..

Intel® Advisor helps you increase performance!

Recommended methodology



5 Steps to Efficient Vectorization - Vector Advisor

(part of Intel® Advisor, Parallel Studio, Cluster Studio 2016)

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization
			Loop Type Why No Vectorization?
[loop in runCForallLambdaLoops]	0.094s	0.094s	Scalar vector dependence prevents vector...
[loop in runCForallLambdaLoops]	0.140s	3.744s	Scalar inner loop was already vectorized
[loop in std::Complex_base<double,struct _C_double_complex>::...	0.031s	0.031s	Vectorized (Body)

Vectorized SSE; SSE2 loop processing Float32; Float64 data type
Peeled loop; loop stats were reordered

[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s

2. Guidance: detect problem and recommend how to fix it

Issue: Peeled/Remainder loop(s) present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials](#), [Utilizing Full Vectors...](#)

Recommendation: Align memory access
Projected maximum performance gain: High
Projection confidence: Medium

Use one of the memory accesses in the source loop does not
...ry access and tell the compiler your memory access is aligned.
byte boundary:

SIZE*sizeof(float), 32);

3. "Accurate" Trip Counts + FLOPs: understand utilization, parallelism granularity & overheads

Total Time	Trip Counts			Iteration Duration	Call Count
	Median	Min	Max		
3.151s	1	1	1	3.1509s	1
0.440s	1	1	1	< 0.0001s	2408000
0.010s	1	1	2	< 0.0001s	207596
0.010s	2	1	9	< 0.0001s	1173619
0.010s	3	1	5	< 0.0001s	1312315

4. Loop-Carried Dependency Analysis

Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	✗ New
P7	Write after read dependency	site2	dqtest2.cpp; idle.h	dqtest2	✗ New

5. Memory Access Patterns Analysis

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_203	runCRawLoops	runCRawLoops.coc1063	RAW:1	No information available	No information available
loop_site_139	runCRawLoops	runCRawLoops.coc622	No information available	39% / 36% / 25%	Mixed strides
loop_site_160	runCRawLoops	runCRawLoops.coc925	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns Correctness Report

ID	Stride	Type	Source	Modules	Alignment
P22	0; 0; 1	Unit stride	runCRawLoops.coc637	lcals.exe	
P23	0; 0	Unit stride	runCRawLoops.coc638	lcals.exe	
P30	-1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801	Variable stride	runCRawLoops.coc628	lcals.exe	

635 j2 = (j2 + 64-1) ;
636 p[ip][0] += y[i2+32];
637 p[ip][1] += z[j2+32];
638 i2 += e[i2+32];
639 j2 += f[j2+32];

626 i1 = 64-1;
627 j1 = 64-1;
628 p[ip][2] += b[j1][i1];

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops▲	Self Time	Total Time			Compiler Vectorization	
					Loop Type	Why No Vectorization?
⊞ [loop in runCForallLambdaLoops]	0.094s	0.094s			Scalar	vector dependence prevents vector...
⊞ [loop in runCForallLambdaLoops]	0.140s	3.744s			Scalar	inner loop was already vectorized
⊞ V [loop in std::complex_base<double,struct _C_double_complex>::f...	0.031s	0.031s			Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations Peeled loop; loop stmts were reordered						
⊞ [loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...			Scalar	nonstandard loop is not a vectoriza ...
⊞ [loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...			Scalar	nonstandard loop is not a vectoriza ...
⊞ [loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	0.234s			Scalar	nonstandard loop is not a vectoriza ...

The Right Data At Your Fingertips

Get all the data you need for high impact vectorization

Filter by which loops are vectorized!

Trip Counts

What prevents vectorization?

Focus on hot loops

What vectorization issues do I have?

Which Vector instructions are being use?

How efficient is the code?

Intel Advisor XE 2016									
Where should I add vectorization and/or threading parallelism?									
Summary Suitability Report Refinement Reports Annotation Report Suitability Report									
Elapsed time: 54.44s Vectorized Not Vectorized FILTER: All Modules All Sources									
Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization?	Vectorized Loops		
							Vecto...	Efficiency	Vector L..
[loop at stl_algo.h:4740 in std::tr ...]		0.170s	0.170s		Scalar	non-vectorizable loop ins ...			
[loop at loopstl.cpp:2449 in s234_]	2 Ineffective peeled/rem ...	0.170s	0.170s	12; 4	Collapse	Collapse	AVX	~100%	4
[loop at loopstl.cpp:2449 in s ...]		0.150s	0.150s	12	Vectorized (Body)		AVX		4
[loop at loopstl.cpp:2449 in s ...]		0.020s	0.020s	4	Remainder				
[loop at loopstl.cpp:7900 in vas_]		0.170s	0.170s	500	Scalar	vectorization possible but ...			4
[loop at loopstl.cpp:3509 in s2 ...]	1 High vector register ...	0.160s	0.160s	12	Expand	Expand	AVX	~69%	8
[loop at loopstl.cpp:3891 in s279_]	2 Ineffective peeled/rem ...	0.150s	0.150s	125; 4	Expand	Expand	AVX	~96%	8
[loop at loopstl.cpp:6249 in s414_]		0.150s	0.150s	12	Expand	Expand	AVX	~100%	4
[loop at numeric.h:247 in std ...]	1 Assumed dependency ...	0.150s	0.150s	49	Scalar	vector dependence preve ...			

All the data in one place

Top Down

Source

Assembly

Recommendations

Compiler Diagnostics

Top Down	Source	Loop Assembly	Assistance	Recommendations	Compiler Diagnostic Details					
Function Call Sites and Loops		Total Time %	Total Time	Self Time	Loop Type	Why No Vectorization?	Vectorized Loops			Instru
							Vecto...	Vector Length	Compiler Estimated Gain	Traits
[-] Total		100.0%	15.043s	0s						
[-] func@0x6b2daccf		100.0%	15.043s	0s						
[-] func@0x6b2dacf0		100.0%	15.043s	0s						
[-] BaseThreadInitThunk		100.0%	15.043s	0s						
[-] _tmainCRTStartup		100.0%	15.043s	0s						
[-] main		99.9%	15.035s	0s						
[+] [loop at Driver.c:145]		99.9%	15.035s	0s	Scalar	loop with function call ...				
[+] printf		0.0%	0.001s	0.0006s						
[+] func@0x10150ef0		0.1%	0.008s	0.0076s						

Background on loop vectorization

A typical vectorized loop consists of

Main vector body

- Fastest among the three!

Optional peel part

- Used for the unaligned references in your loop. Uses Scalar or slower vector

Remainder part

- Due to the number of iterations (trip count) not being divisible by vector length. Uses Scalar or slower vector.

Larger vector register means more iterations in peel/remainder

- Make sure you Align your data!
- Make the number of iterations divisible by the vector length!



This is where we want our loops to be executing!

Efficiently Vectorize your code

Intel Advisor – Vectorization Advisor

Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 54.44s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization?	Vectorized Loops
							Vecto... Efficiency Vector L...
[loop at stl_algo.h:4740 in std::tr...		0.170s	0.170s		Scalar	non-vectorizable loop ins ...	
[loop at loopstl.cpp:2449 in s234_]	2 Ineffective peeled/rem...	0.170s	0.170s	12; 4	Collapse	Collapse	AVX ~100% 4
[loop at loopstl.cpp:2449 in s...		0.150s	0.150s	12	Vectorized (Body)		AVX 4
[loop at loopstl.cpp:2449 in s...		0.020s	0.020s	4	Remainder		
[loop at loopstl.cpp:7900 in vas_]		0.170s	0.170s	500	Scalar	vectorization possible but...	4
[loop at loopstl.cpp:3509 in s2...	1 High vector register ...	0.160s	0.160s	12	Expand	Expand	AVX ~69% 8
[loop at loopstl.cpp:3891 in s279_]	2 Ineffective peeled/rem...	0.150s	0.150s	125; 4	Expand	Expand	AVX ~96% 8
[loop at loopstl.cpp:6249 in s414_]		0.150s	0.150s	12	Expand	Expand	AVX ~100% 4
[loop at stl_numeric.h:247 in std...	1 Assumed dependency...	0.150s	0.150s	49	Scalar	vector dependence preve ...	

Top Down Source Loop Assembly Assistance Recommendations Compiler Diagnostic Details


File: loopstl.cpp:3509 s273_

Line	Source	Total Time	%	Loop Time	%
3504	forttime_ (&t1);				
3505	i_1 = *ntimes;				
3506	for (nl = 1; nl <= i_1; ++nl)	0.010s		0.200s	
	[loop at loopstl.cpp:3506 in s273_] Scalar Loop. Not vectorized: inner loop was already vectorized No loop transformations were applied				
3507	{				
3508	i_2 = *n;				
3509	for (i_ = 1; i_ <= i_2; ++i_)	0.010s		0.160s	
	[loop at loopstl.cpp:3509 in s273_] Vectorized AVX Loop processing Float32; Float64; Int32 data type(s) having Inserts; Extracts; Masked St...				
	Selected (Total Time):	0.010s			


1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time			Compiler Vectorization	
					Loop Type	Why No Vectorization?
[loop in runCForallLambdaLoops]	0.094s	0.094s			Scalar	vector dependence prevents vector...
[loop in runCForallLambdaLoops]	0.140s	3.744s			Scalar	inner loop was already vectorized
[loop in std::complex_base<double,struct _C_double_complex>::d...]	0.031s	0.031s			Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations						
Peeled loop; loop stats were reordered						
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...			Scalar	nonstandard loop is not a vectoriza...
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...			Scalar	nonstandard loop is not a vectoriza...
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	0.234s			Scalar	nonstandard loop is not a vectoriza...


2. Guidance: detect problem and recommend how to fix it


2

Issue: Peeled/Remainder loop(s) present


8

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials](#), [Utilizing Full Vectors...](#)


Recommendation: Align memory access

Projected maximum performance gain: High
Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary:

```
float *array;
array = (float *)_mm_malloc(ARRAY_SIZE*sizeof(float), 32);

// Somewhere else
__assume_aligned(array, 32);
// Use array in loop
```

Get Specific Advice For Improving Vectorization

Intel® Advisor – Vectorization Advisor

Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 8,81s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?	Vectorized Loops		
						Vecto...	Estim...	Vector Len
[loop at market...]			11,460s	Scalar				
[loop at arena.cpp:88 in tbb::tbb::...]		0,000s	11,460s	Scalar				
[loop at fractal.cpp:179 in <lambda1>::op ...]	5 Ineffective ...	0,000s	2,022s	Collapse	Collapse			
[loop at fractal.cpp:179 in <lambda1>::o ...]	2 Data type co ...	0,000s	2,022s	Remainder				

Click to see recommendation

Top Down Source Loop Assembly Assistance Recommendations Compiler Diagnostic Details

Issue: Ineffective peeled/remainder loop(s) present

All or some [source loop](#) iterations are not executing in the [loop body](#). Improve performance by moving source loop iterations from [peeled](#) / [remainder](#) loops to the loop body.

Disable unrolling

The [trip count](#) after loop unrolling is too small compared to the [unroll factor](#) using a [directive](#).

ICL/ICC/ICPC Directive	IFORT Directive
#pragma nounroll	!DIR\$ NOUNROLL
#pragma unroll	!DIR\$ UNROLL

Read More:

- [User and Reference Guide for the Intel C++ Compiler 15.0](#) > [Compiler Reference](#) > [Pragmas](#) > [Intel-specific Pragma Reference](#) > [unroll/nounroll](#).

Advisor shows hints to move iterations to vector body.

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time		Compiler Vectorization	
				Loop Type	Why No Vectorization?
[loop in runCForAllLambdaLoops]	0.094s	0.094s		Scalar	vector dependence prevents vector...
[loop in runCForAllLambdaLoops]	0.140s	3.744s		Scalar	inner loop was already vectorized
[loop in std::complex_base<double,struct _C_double_complex>::ci...	0.031s	0.031s		Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations					
Peeled loop; loop starts were reordered					
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	0.000s			
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	0.000s			
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	0.000s			

2. Guidance: detect problem and recommend how to fix it

- Issue: Peeled/Remainder loop(s) present**
- All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials, Utilizing Full Vectors...](#)
- Recommendation: Align memory access**
- Projected maximum performance gain: High
Projection confidence: Medium
- The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary:

```
..._SIZE*sizeof(float), 32);
```

3. “Accurate” Trip Counts + FLOPs: understand utilization, parallelism granularity & overheads

Total Time	Trip Counts				
	Median	Min	Max	Iteration Duration	Call Count
3.151s	1	1	1	3.1509s	1
0.440s	1	1	1	< 0.0001s	2408000
0.010s	1	1	2	< 0.0001s	207596
0.010s	2	1	9	< 0.0001s	1173619
0.010s	3	1	5	< 0.0001s	1312315

Critical Data Made Easy

Loop Trip Counts

Knowing the time spent in a loop is not enough!

Intel Advisor XE 2016

Where should I add vectorization and/or threading parallelism?

Summary Survey Report Refinement Reports Annotation Report Suitability

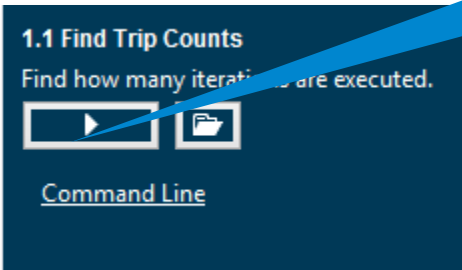
Program time: 12.82s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	Self Time	Total Time			Trip Counts			Call Count	Compiler Vectorization	
					Median	Min	Max		Loop Type	Why No Vectorization
[loop at Multiply.c:53 in matvec]	11.898s	11.898s		1					Collapse	Collapse
[loop at Multiply.c:53 in matvec]	11.851s	11.851s		1	101	101	101	12000000	Vectorized (Body)	vector dependence p
[loop at Multiply.c:53 in matvec]	0.047s	0.047s			3	3	3	1000000	Vectorized (Body)	
[loop at Multiply.c:53 in matvec]	0.413s	0.413s			101	101	101	2000000	Scalar	
[loop at Multiply.c:45 in matvec]	0.109s	12.373s		1					Expand	Expand
[loop at Driver.c:146 in main]	0.016s	12.483s		1	1000000	1000000	1000000	1	Scalar	vector dependence p

Check actual trip counts

Loop is iterating 101 times but called > million times

Since the loop is called so many times it would be a big win if we can get it to vectorize.



1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time			Compiler Vectorization	
					Loop Type	Why No Vectorization?
[loop in runCForAllLambdaLoops]	0.094s	0.094s			Scalar	vector dependence prevents vector...
[loop in runCForAllLambdaLoops]	0.140s	3.744s			Scalar	inner loop was already vectorized
[loop in std::complex_base<double,struct _C_double_complex>::ci...	0.031s	0.031s			Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations						
Peeled loop; loop stats were reordered						
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000					
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000					
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000					

2. Guidance: detect problem and recommend how to fix it

Issue: Peeled/Remainder loop(s) present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials](#), [Utilizing Full Vectors...](#)

Recommendation: Align memory access

Projected maximum performance gain: High
Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary:

3. "Accurate" Trip Counts + FLOPs: understand utilization, parallelism granularity & overheads

Total Time	Trip Counts			Iteration Duration	Call Count
	Median	Min	Max		
3.151s	1	1	1	3.1509s	1
< 0.0001s				2408000	
< 0.0001s				207596	
< 0.0001s				1173619	
< 0.0001s				1312315	

4. Loop-Carried Dependency Analysis

Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	✗ New
P7	Write after read dependency	site2	dqtest2.cpp; idle.h	dqtest2	✗ New

Is It Safe to Vectorize?

Loop-carried dependencies analysis verifies correctness

« Ad Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Program time: 12.82s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	Self Time	Total Time			Trip Counts	Compiler Vectorization	
						Loop Type	Why No Vectorization?
i> [loop at Multiply.c:53 in matvec]	0.047s	0.047s			3	Vectorized (Body)	
i> [loop at Multiply.c:53 in matvec]	0.413s	0.413s			101	Scalar	
⊞ [loop at Multiply.c:45 in matvec]	0.109s	12.373s		1		Collapse	Collapse
i> [loop at Multiply.c:45 in matvec]	0.078s	11.930s			12	Vectorized (Body)	
i> [loop at Multiply.c:45 in matvec]	0.031s	0.444s			2	Remainder	
⊞ [loop at Driver.c:146 in main]	0.016s	12.483s	✓	1	1000000	Scalar	vector dependence prevents vectoriza...

2.1 Check Correctness

Identify and explore loop-carried dependencies for marked loops. Fix the reported problems.

▶

Command Line

Select loop for
Correct
Analysis and
press play!

Vector Dependence
prevents
Vectorization!

Data Dependencies – Tough Problem #1

Is it safe to force the compiler to vectorize?

Data dependencies

```
for (i=0;i<N;i++)          // Loop carried dependencies!  
  
    A[i] = A[i-1]*C[i];    // Need the ability to check if it  
  
                           // it is safe to force the compiler
```

Issue: Assumed dependency present

The compiler assumed there is an anti-dependency (Write after read – WAR) or true dependency (Read after write – RAW) in the loop. Improve performance by investigating the assumption and handling accordingly.

➤ Enable vectorization

Potential performance gain: Information not available until Beta Update release

Confidence this recommendation applies to your code: Information not available until Beta Update release

The Correctness analysis shows there is no real dependency in the loop for the given workload. Tell the compiler it is safe to vectorize using the `restrict` keyword or a [directive](#).

ICL/ICC/ICPC Directive	IFORT Directive	Outcome
#pragma simd or #pragma omp simd	!DIR\$ SIMD or !SOMP SIMD	Ignores all dependencies in the loop
#pragma ivdep	!DIR\$ IVDEP	Ignores only vector dependencies (which is safest)

Read More:

- [User and Reference Guide for the Intel C++ Compiler 15.0](#) > **Compiler Reference** > **Pragmas** > **Intel-specific**
Pragma Reference >
 - `ivdep`
 - `omp simd`

Correctness – Is It Safe to Vectorize?

Loop-carried dependencies analysis

Received recommendations to force vectorization of a loop:

1. Mark-up the loop and check for the presence of REAL dependencies
2. Explore dependencies in more details with code snippets

In this example 3 dependencies were detected

- RAW – Read After Write
- WAR – Write After Read
- WAW – Write After Write

This is NOT a good candidate to force vectorization!

The screenshot displays the Intel Advisor interface for analyzing loop-carried dependencies. At the top, a summary bar shows the status of various reports. Below this, a table provides an overview of the analysis for 'loop_site_6'. A blue callout box labeled 'Detected dependencies' points to the 'Problems and Messages' section, which lists three detected dependencies: P1 (Parallel site information), P3 (Read after write dependency), and P4 (Write after write dependency). Below this, a detailed view of the 'Write after read dependency' is shown, highlighting specific source lines (20-24) where both read and write accesses to memory locations are detected, indicating a RAW dependency. A blue callout box labeled 'Source lines with Read and Write accesses detected' points to these highlighted lines.

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_6	main	main.cpp:13	RAW:1 WAR:1 WAW:1	91% / 0% / 9%	Mixed strides

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	loop_site_6	main.cpp	test_1.exe	✓ Not a problem
P3	Read after write dependency	loop_site_6	crtexe.c; main.cpp	test_1.exe	New
P4	Write after write dependency	loop_site_6	crtexe.c; main.cpp	test_1.exe	New
P5	Write after read dependency	loop_site_6	crtexe.c; main.cpp	test_1.exe	New

ID	Description	Source	Function	Module	State
X17	Read	main.cpp:22	main	test_1.exe	New
20 k += a[9];					
21 k *= a[8];					
22 k -= a[7];					
23 k += a[6];					
24 k *= a[5];					
X18	Read	main.cpp:23	main	test_1.exe	New
21 k *= a[8];					
22 k -= a[7];					
23 k += a[6];					

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time		Compiler Vectorization	
				Loop Type	Why No Vectorization?
[loop in runCForallLambdaLoops]	0.094s	0.094s		Scalar	vector dependence prevents vector...
[loop in runCForallLambdaLoops]	0.140s	3.744s		Scalar	inner loop was already vectorized
[loop in std::complex_base<double,struct _C_double_complex>::ci...	0.031s	0.031s		Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations					
Peeled loop; loop stats were reordered					
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000				
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000				
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000				

3. “Accurate” Trip Counts + FLOPs: understand utilization, parallelism granularity & overheads

Total Time	Trip Counts			Iteration Duration	Call Count
	Median	Min	Max		
3.151s	1	1	1	3.1509s	1

2. Guidance: detect problem and recommend how to fix it

Issue: Peeled/Remainder loop(s) present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials, Utilizing Full Vectors...](#)

Recommendation: Align memory access

Projected maximum performance gain: High
Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary:

```
...SIZE*sizeof(float), 32);
```

4. Loop-Carried Dependency Analysis

Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	✗ New
P7	Write after read dependency	site2	dqtest2.cpp, idle.h	dqtest2	✗ New

5. Memory Access Patterns Analysis

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_203	runCRawLoops	runCRawLoops.cxx1063	RAW:1	No information available	No information available
loop_site_139	runCRawLoops	runCRawLoops.cxx622	No information available	39% / 36% / 25%	Mixed strides
loop_site_160	runCRawLoops	runCRawLoops.cxx925	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns		Correctness Report			
ID	Stride	Type	Source	Modules	Alignment
P22	0; 0; 1	Unit stride	runCRawLoops.cxx637	lcals.exe	
<pre>635 j2 = (j2 < 64-1) ; 636 p[ip][0] += y[i2+32]; 637 p[ip][1] += z[j2+32]; 638 i2 += e[i2+32]; 639 j2 += f[j2+32];</pre>					
P23	0; 0	Unit stride	runCRawLoops.cxx638	lcals.exe	
P30	-1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801	Variable stride	runCRawLoops.cxx628	lcals.exe	
<pre>626 i1 <= 64-1; 627 j1 <= 64-1; 628 p[ip][2] += b[j1][i1];</pre>					

Non-Contiguous Memory – Tough Problem #2

Potential to vectorize but may be inefficient

- Non-unit strided access to arrays

```
for (i=0;i<N;i+=2) //Incrementing "i" by 2 is not unit stride  
  
    //We need a way to check how we are  
  
    //accessing memory.
```

- Indirect reference in a loop

```
for (i=0;i<N;i++)  
  
    A[B[i]] = C[i]*D[i]; //We have to decode B[i] to find out  
                        //which element of A to reference
```

Improve Vectorization

Memory Access pattern analysis

Where should I add vectorization and/or threading parallelism?

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 8,52s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops				Loop Type	Why No Vectorization?
[loop at fractal.cpp:179 in <lambda1>::op ...]				Collapse	Collapse
[loop at fractal.cpp:179 in <lambda1>::o ...]	<input checked="" type="checkbox"/>	4 Serialized use ...	0,013s 11,281s	Vectorized (Body)	
[loop at fractal.cpp:179 in <lambda1>::o ...]	<input checked="" type="checkbox"/>	2 Data type co ...	0,000s 0,163s	Peeled	
[loop at fractal.cpp:179 in <lambda1>::o ...]	<input checked="" type="checkbox"/>	2 Data type co ...	0,000s 0,576s	Remainder	
[loop at fractal.cpp:177 in <lambda1>::oper ...]	<input type="checkbox"/>	2 Data type co ...	0,010s 12,030s	Scalar	

Select loops of interest

2.2 Check Memory Access Patterns

Identify and explore complex memory accesses for marked loops. Fix the reported problems.



Command Line

Run Memory Access Patterns analysis, just to check how memory is used in the loop and the called function

Find vector optimization opportunities

Memory Access pattern analysis

Stride distribution

Check memory access patterns in your application

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_79	operator()	fractal.cpp:179	No information available	100% / < 1,0000% / ...	Mixed strides
loop_site_93	operator()	fractal.cpp:179	No information available	100% / 0% / 0%	All unit strides
loop_site_94	operator()	fractal.cpp:179	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns Report

ID	Stride	Type	File	Line	Executable
P18	0	Unit stride	fractal.cpp:66		fractal.exe
P21	0	Unit stride	fractal.cpp:66		fractal.exe
P24	0	Unit stride	fractal.cpp:68		fractal.exe
P27	0	Unit stride	fractal.cpp:69		fractal.exe
P30	0	Unit stride	fractal.cpp:74		fractal.exe

```
64 color_t color;  
65  
66 fx0 = x0 - size_x / 2.0f;  
67 fy0 = y0 - size_y / 2.0f;  
68 fx0 = fx0 / magn + cx;  
69 fy0 = fy0 / magn + cy;  
70
```

All memory accesses are uniform, with zero unit stride, so the same data is read in each iteration

We can therefore declare this function using the omp syntax: `pragma omp declare simd uniform(x0`

Advisor 2017 Update 2 Features

- 1. Cache aware Roofline
- 2. Improved Trip Counts and FLOPS
 - 1. **Call Count** metric for functions
- 3. Filtering by module
- 4. Re-finalization
- 5. Dynamic Instruction Mixes

FLOPS And AVX-512 Mask Usage			Vectorized Loops			Instruction Set Analysis	
GFLOPS	AI	Mask Utilization	Vector...	Efficiency	Gain Estim...	VL (...)	Traits
2,080	0,1243	100,0%	AVX512	~100%	17.50x	16; 8	FMA; Mask Manipulations
0,856	0,0809	91,7%	AVX512	~100%	17.69x	16; 8	FMA; Mask Manipulations
0,455	0,1398	89,6%	AVX512	~100%	14.41x	16; 8	FMA; Mask Manipulations
0,234	0,1472	100,0%					Appr. Reciprocals(AVX-512ER); Expon...
0,148	0,1429						FMA
0,095	0,0722	40,1%					FMA; Square Roots; Type Conversions
0,091	0,0208						FMA
0,074	0,1429						FMA

- 1. -report survey -mix

☒ 1.1 Find Trip Counts and FLOPS

- ☒ Trip Counts
- ☒ FLOPS

Vectorization WorkflowThreading Workflow

OFFBatch mode

Run Roofline

▶Collect

1. Survey Target

▶Collect

1.1 Find Trip Counts and FLOPS

Re-finalize Survey

roofline_demo_samples - Project Properties

Analysis TargetBinary/Symbol SearchSource Search

Survey Analysis Types

- Survey Hotspots Ana
- Survey Trip Count An
- Suitability Analysis

Refinement Analysis Typ

- Dependencies Analy
- Memory Access Patte

Survey Launch Application

Specify and configure the application executable (target) to analyze. Press F1 for more details.

Modules:

☐ Include only the following module(s)

☒ Exclude the following module(s)

Specify application (or child application) module(s) to include in or exclude from inspection.

Modify...

Vectorized loops with high efficiency

Ad Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

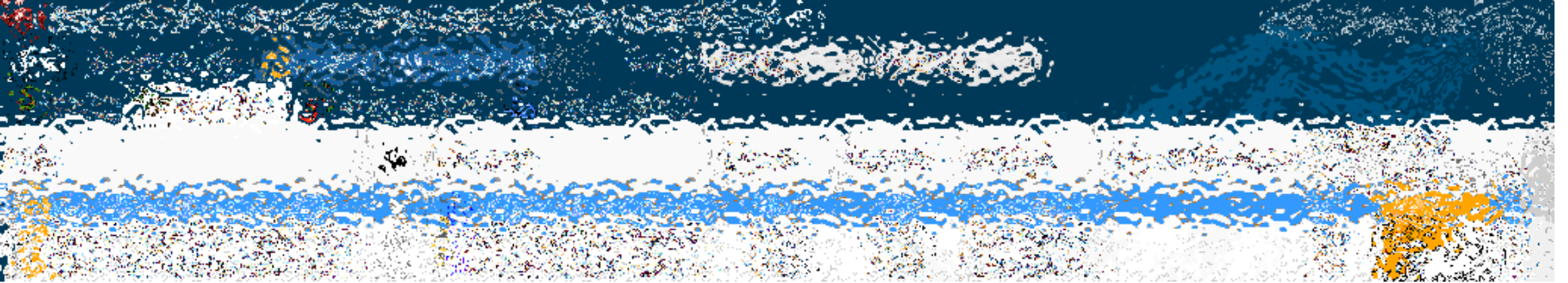
Elapsed time: 12.61s Vectorized Not Vectorized FILTER: All Modules All Sources

Summary Survey Report Refinement Reports Annotation Report

Loops		Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?	Vectorized Loops		
							Vect...	Efficiency	Gai
[loop in fGetEquilibriumF at lbpSUB.c ...]	<input type="checkbox"/>	1 Data type conversions present	0.355s	0.355s	Vectorized (Bo...		AVX	100%	5.03
[loop in fCollisionBGK at lbpBGK.cpp: ...]	<input type="checkbox"/>	1 Ineffective peeled/remainder ...	0.047s	0.047s	Vectorized (Re...		AVX	67%	1.34
[loop in fGetFracSite at lbpGET.cpp:19 ...]	<input type="checkbox"/>	1 Possible inefficient memory a ..	0.020s	0.020s	Vectorized Vers...		AVX	19%	1.56

Are we done??..

Vectorized loops with high efficiency



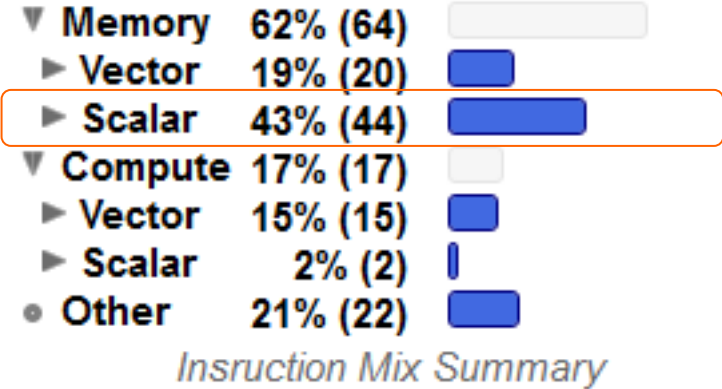
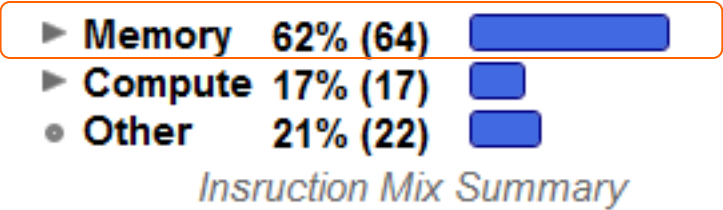
...It depends.

If code is not SIMD bound,
then Speedup \leq Vectorization Gain

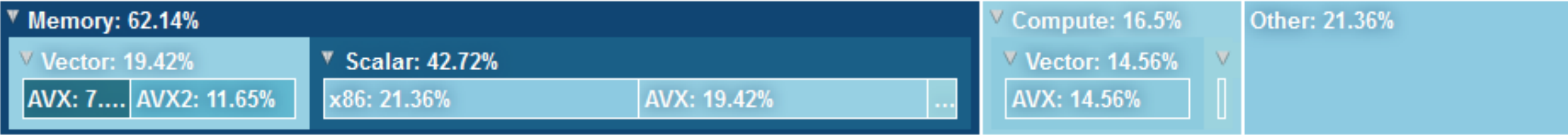
In addition (instead of) VPU-bound
code could be Memory Bound

Am I bound by VPU/CPU or by Memory?

Quick and Dirty check with **Survey Loop Analytics**.

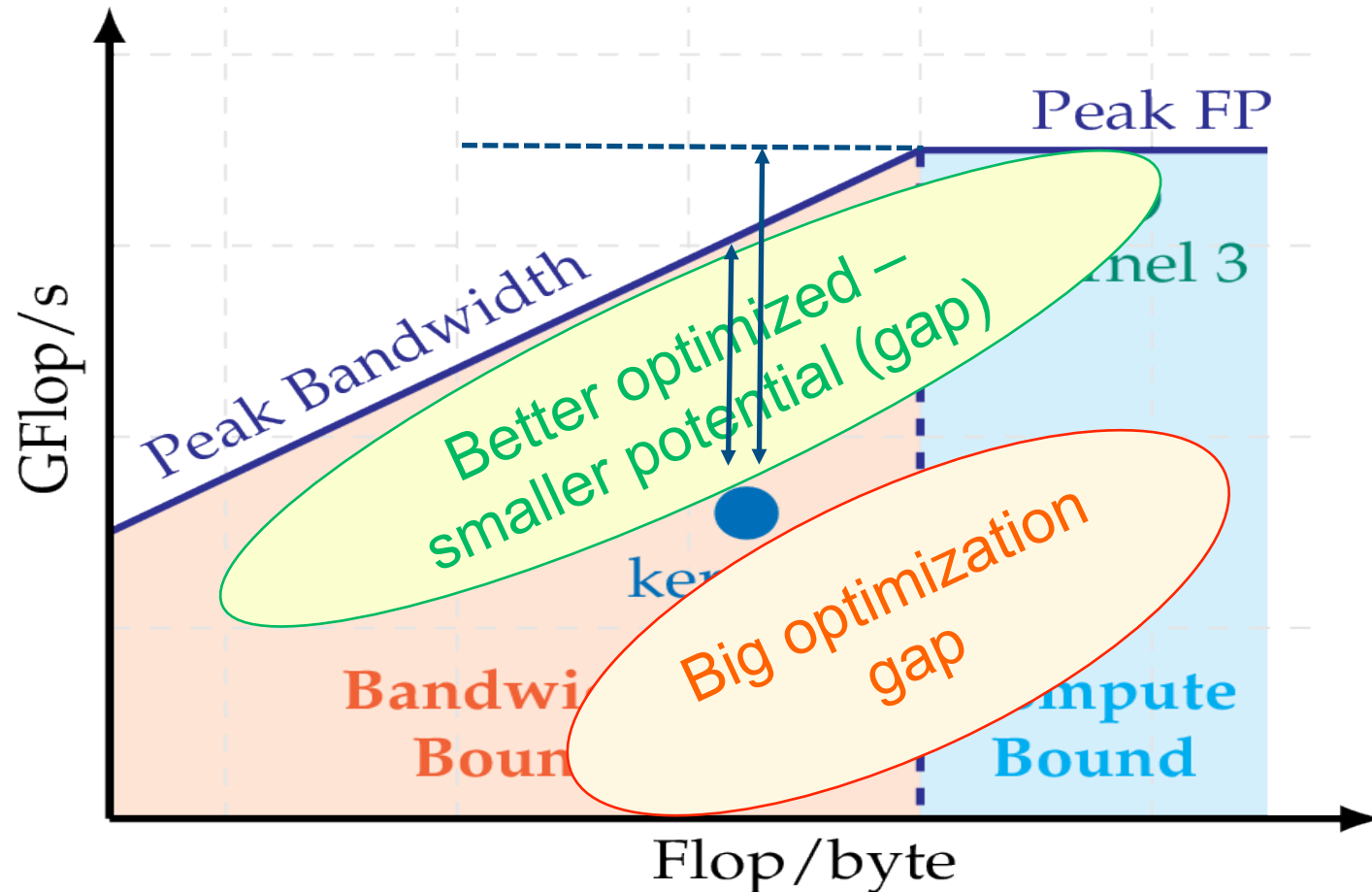


The types of instructions in your loop will be a rough indicator of whether you are doing more memory or computational work



Am I bound by VPU/CPU or by Memory?

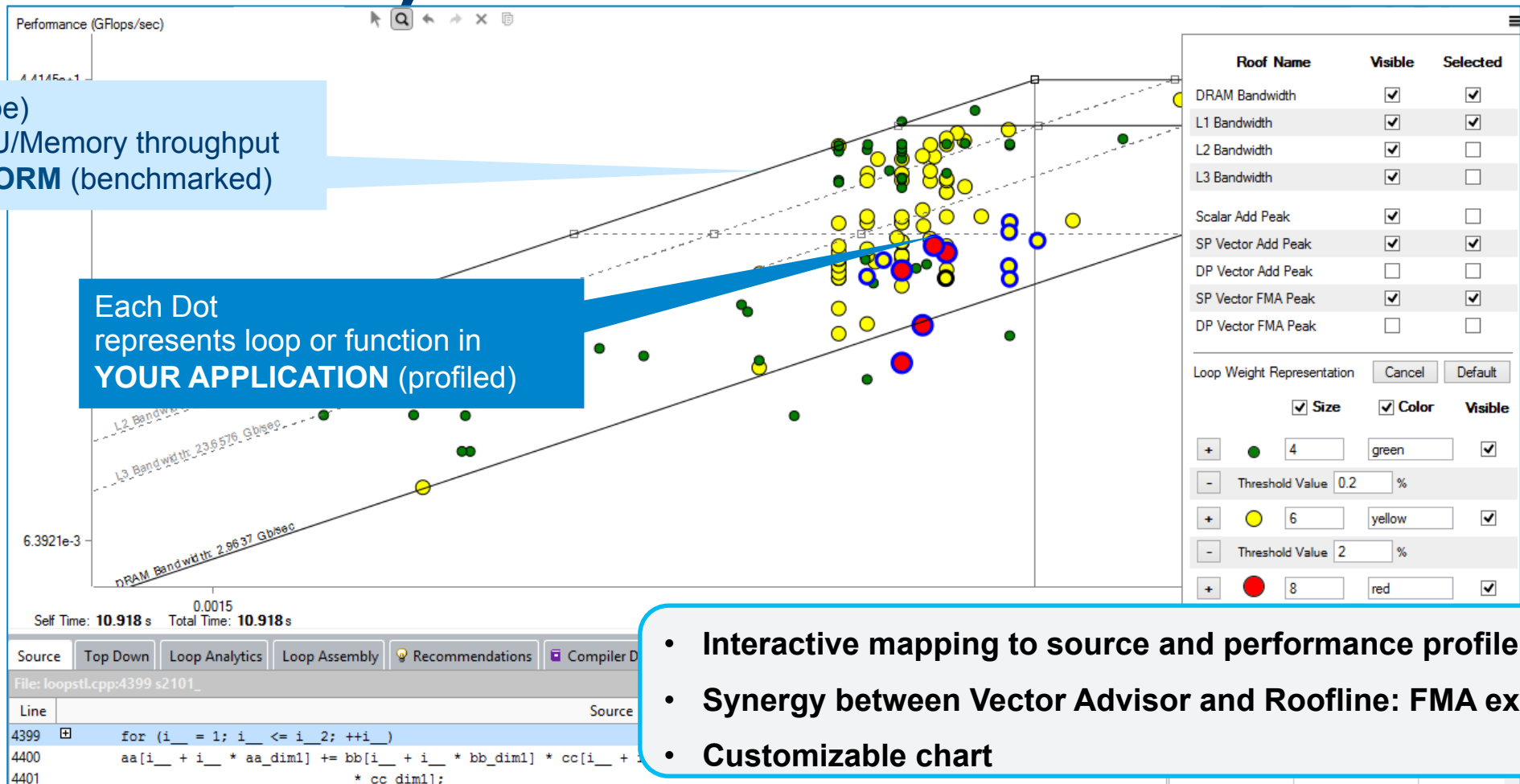
ROOFLINE ANALYSIS



Roofline Automation in Intel® (Vectorization) Advisor 2017

Each Roof (slope)
Gives peak CPU/Memory throughput
of your **PLATFORM** (benchmarked)

Each Dot
represents loop or function in
YOUR APPLICATION (profiled)



- Interactive mapping to source and performance profile
- Synergy between Vector Advisor and Roofline: FMA example
- Customizable chart

Advisor Roofline: under the hood

Roofline application profile:

Axis Y: **FLOP/S** = **#FLOP** (mask aware) / **#Seconds**

Axis X: **AI** = **#FLOP** / **#Bytes**

Seconds

User-mode **sampling**

Root access not needed

Roofs

Microbenchmarks

Actual peak for the current configuration

Performance = Flops/seconds

#FLOP

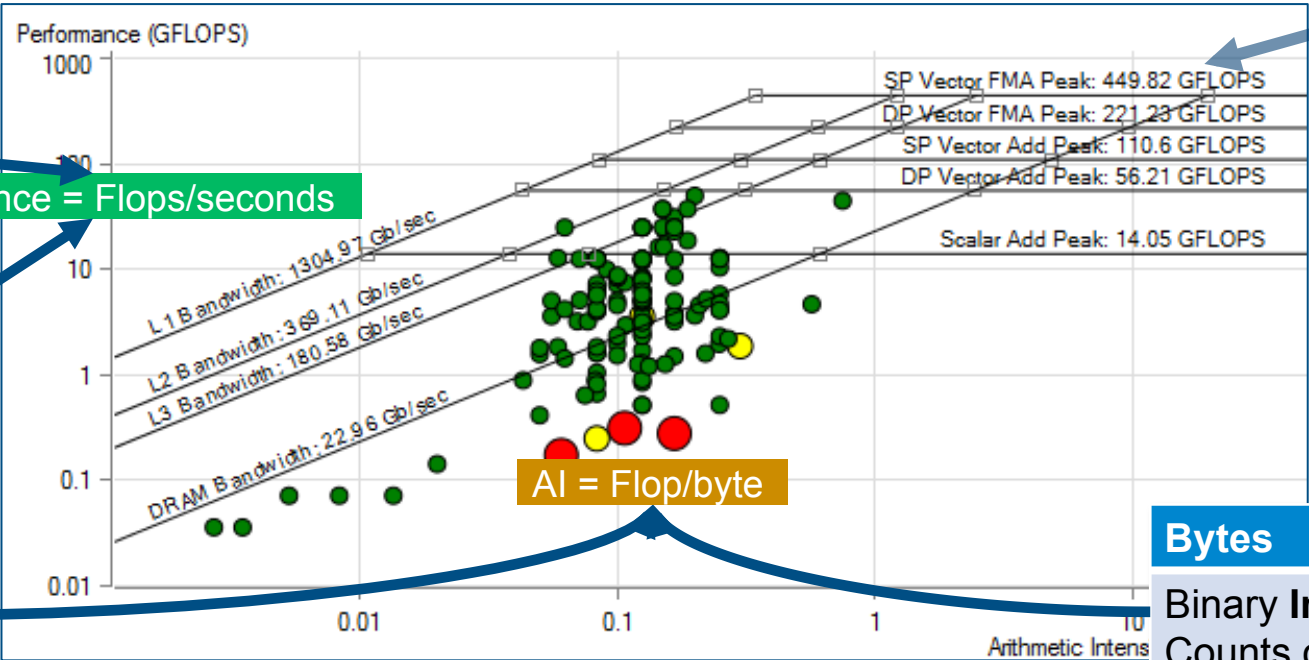
Binary **Instrumentation**

Does not rely on CPU counters

Bytes

Binary **Instrumentation**

Counts operands size (not cachelines)



Getting Roofline in Advisor

1. Survey Target

▶ Collect

▶

▶



▶

1.1 Find Trip Counts and FLOPS

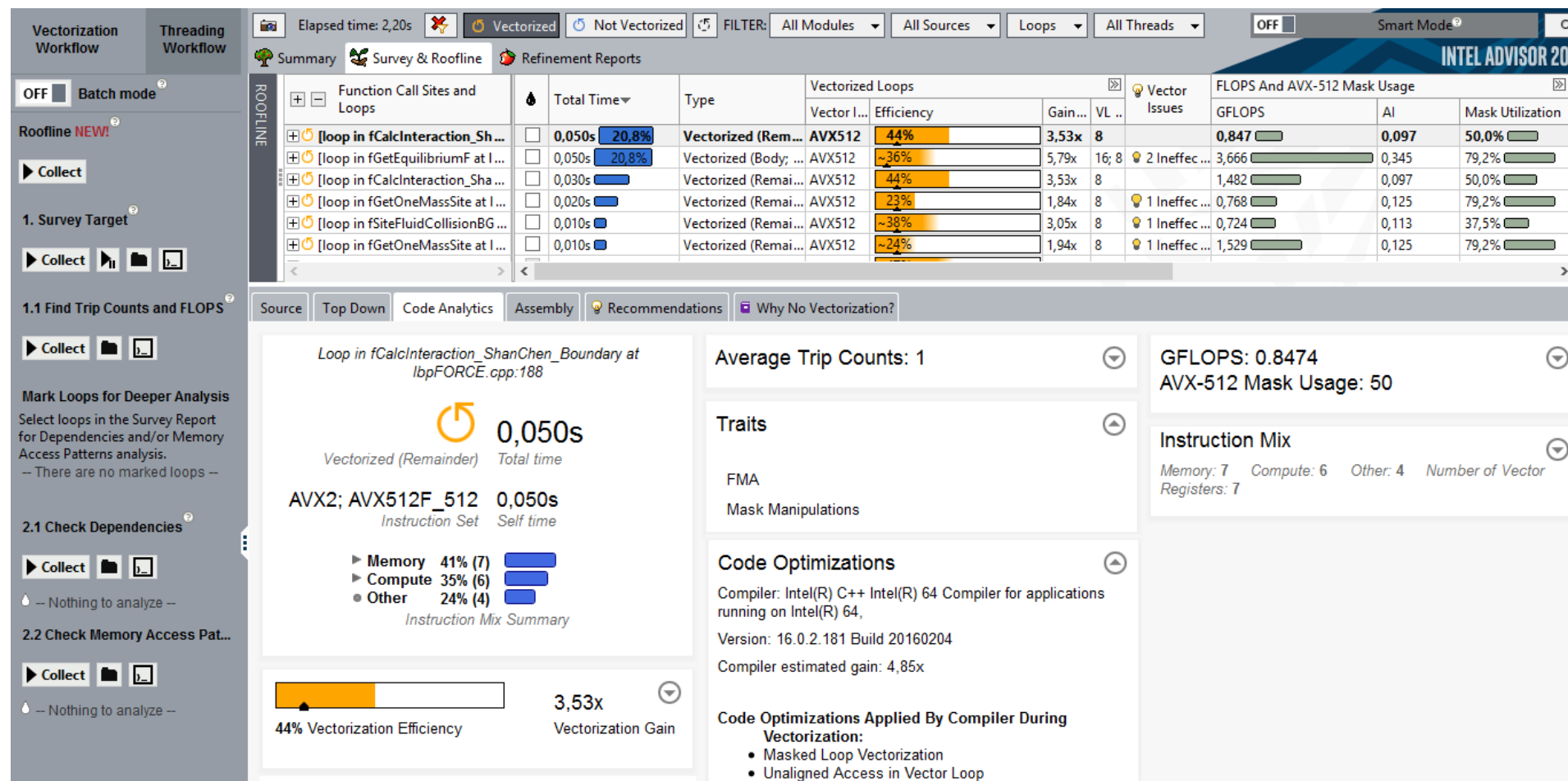
▶ Collect

▶

▶

FLOP/S = #FLOP/Seconds	Seconds	#FLOP Count - Mask Utilization - #Bytes
Step 1: Survey <ul style="list-style-type: none">- Non intrusive. <i>Representative</i>- Output: Seconds (+much more)		
Step 2: FLOPS <ul style="list-style-type: none">- Precise, instrumentation based- Physically count Num-Instructions- Output: #FLOP, #Bytes		

Survey+FLOPs Report on AVX-512: FLOP/s, Bytes and AI, Masks and Efficiency



General efficiency (*FLOPS*) vs. VPU-centric efficiency (Vector Efficiency)

Function Call Sites and Loops	Total Time	Type	Vectorized Loops				Vector Issues	FLOPS And AVX-512 Mask Usage		
			Vector I...	Efficiency	Gain...	VL ..		GFLOPS	AI	Mask Utilization
[+] [loop in fCalcInteraction_Sh...	0,050s 20,8%	Vectorized (Rem...	AVX512	44%	3,53x	8		0,847	0,097	50,0%
[+] [loop in fGetEquilibriumF at I...	0,050s 20,8%	Vectorized (Body; ...	AVX512	~36%	5,79x	16; 8	2 Ineffec...	3,666	0,345	79,2%
		Vectorized (Remai...	AVX512	44%	3,53x	8		1,482	0,097	50,0%
		Vectorized (Remai...	AVX512	23%	1,84x	8	1 Ineffec...	0,768	0,125	79,2%
		Vectorized (Remai...	AVX512	~38%	3,05x	8	1 Ineffec...	0,724	0,113	37,5%
		Vectorized (Remai...	AVX512	~24%	1,94x	8	1 Ineffec...	1,529	0,125	79,2%

High Vector Efficiency
Low FLOPS

Function Call Sites and Loops	Total Time	Type	Vectorized Loops				Vector Issues	FLOPS And AVX-512 Mask Usage		
			Vector I...	Efficiency	Gain...	VL ..		GFLOPS	AI	Mask Utilization
[+] [loop in fCalcInteraction_Sh...	0,050s 20,8%	Vectorized (Remai...	AVX512	44%	3,53x	8		0,847	0,097	50,0%
[+] [loop in fGetEquilibriumF at I...	0,050s 20,8%	Vectorized (Body; ...	AVX512	~36%	5,79x	16; 8	2 Ineffec...	3,666	0,345	79,2%
		Vectorized (Remai...	AVX512	44%	3,53x	8		1,482	0,097	50,0%
		Vectorized (Remai...	AVX512	23%	1,84x	8	1 Ineffec...	0,768	0,125	79,2%
		Vectorized (Remai...	AVX512	~38%	3,05x	8	1 Ineffec...	0,724	0,113	37,5%
		Vectorized (Remai...	AVX512	~24%	1,94x	8	1 Ineffec...	1,529	0,125	79,2%

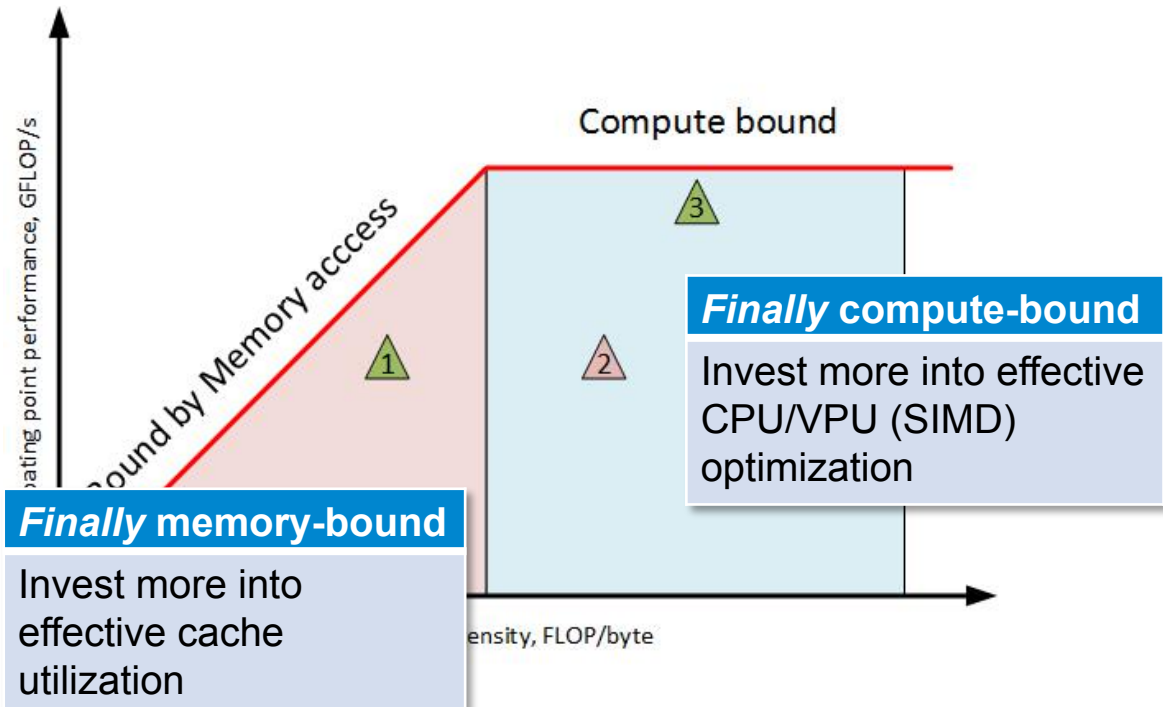
Low Vector Efficiency
High FLOPS

Interpreting Roofline Data: advanced ROI analysis.

Final Limits

(assuming perfect optimization)

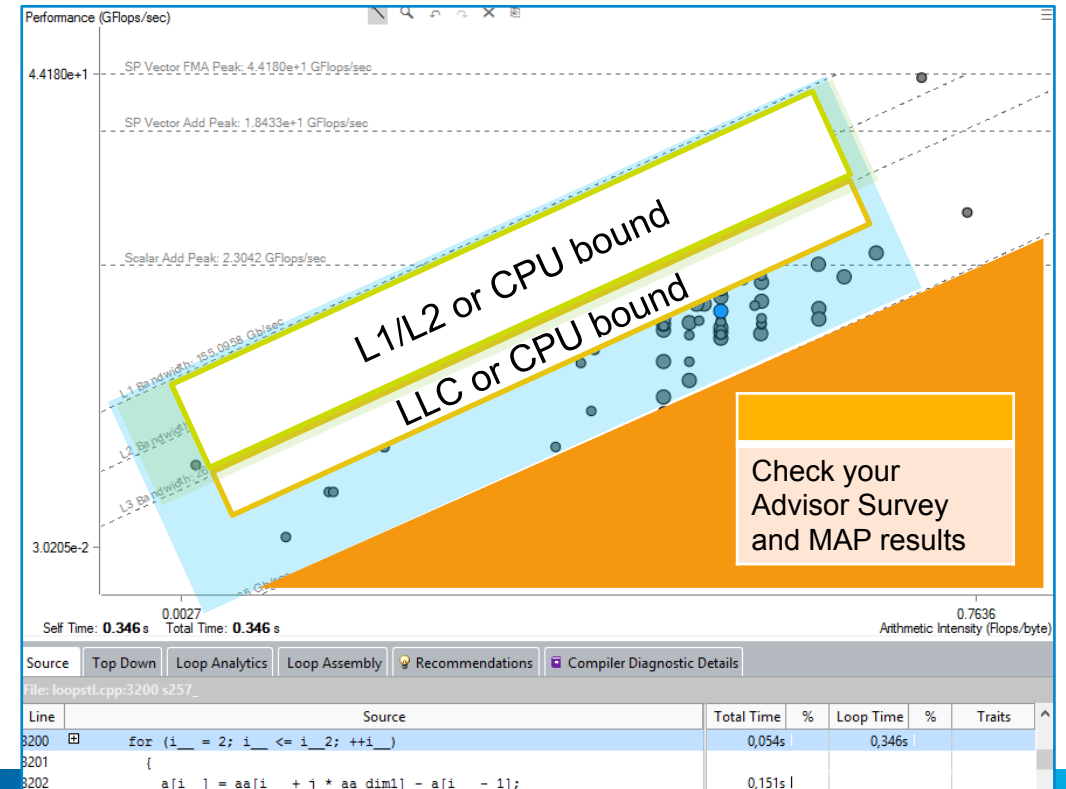
Long-term ROI, optimization strategy



Current Limits

(what are my current bottlenecks)

Next step, optimization tactics



Command line usage

- `advixe-cl -collect survey -project-dir ./your_project -no-auto-finalize -search-dir src=./srcPath -search-dir bin=./binPath -- ./yourExecutable`
- `advixe-cl -collect map -mark-up-list=10,12,15 -project-dir ./your_project -search-dir bin=./binPath -search-dir src=./srcPath -- ./yourExecutable`
- `advixe-cl -collect dependencies --project-dir ./your_project --loops="loop-height=0,total-time>2" -- ./yourExecutable`
- `advixe-cl -report survey -project-dir ./yourProject --search-dir src:r=./src`
- `advixe-cl -collect survey -module-filter-mode=include -module-filter=AnalyzeMyApp.exe,AnalyzeThisToo.dll -project-dir MyProject -- AnalyzeMyApp.exe`
- `advixe-cl -collect survey -module-filter-mode=exclude -module-filter=DoNotAnalyze.so -project-dir MyProject -- MyApplication`

Roofline access and how-to

- For 2017 Update 1

(!) Requires env variable set before running command line or GUI:

```
export ADVIXE_EXPERIMENTAL=roofline
```

- Starting from 2017 Update 2

Just available by default

Roofline access and how-to command line example

(optional) > `source advixe-vars.sh`

(optional) > `export ADVIXE_EXPERIMENTAL=roofline`

> `advixe-cl --collect survey -no-auto-finalize --project-dir ./your_project`
`-- <your-executable-with-parameters>`

> `advixe-cl --collect tripcounts -flops-and-masks --project-dir ./`
`your_project -- <your-executable-with-parameters>`

> `advixe-gui ./your_project`

FLOP/S =
#FLOP/Seconds

1st pass
Obtain "Seconds"
1.1x overhead

2nd pass
Obtain #FLOP count:
3x-5x overhead

Launch GUI

